In the Specification

Please replace the paragraph beginning at line 15 on page 10 with the following marked up paragraph:

> Fig. 5C(1) is a block diagram illustrating the arrangement of detailed views for a database schema employed by the order scheduling system.

Please add the following new paragraph after the paragraph ending at line16 on page 10:

> Figs. 5C(2)-5C(4) are block diagrams illustrating detailed views of a database schema employed by the order scheduling system.

Please replace the paragraph beginning at line 27 on page 18 with the following marked up paragraph:

> Figs. 5C(2)-5C(4) illustrate~~s~~ construction of the database ~~550~~ in further detail. The database itself may be implemented as an SQL-based relational database, such as an Oracle database (e.g., in Oracle 8i, available from Oracle Corporation of Redwood Shores, CA). In particular, the figures demonstrate~~s~~ a detailed views of a database schema employed for the database ~~550~~ in the currently-preferred embodiment. Fig. 5C(1) provides an overview of the arrangement of the detailed views illustrated in Figs. 5C(2)-5C(4). The following database tables are of particular interest in the database schema:

Please replace the computer program listing beginning on page 24, line 14 with the following paragraph.

```
// Return a list of fulfillers ordered by those closest to this
// zipCode
public Vector byProximity (String zipCode) {
  int zoneOfZipCode;
  Vector vectorOfFulfillers = new Vector();
  Vector fulfillersTmp;
  int step = 1;
  boolean keepGoing;
  int i;

  // The first digit of a zip code is the "national area" of the
  // country.
  // The areas are:
  //   0 Northeast              5 Midwest
  //   1 NewYork                6 Plains
//   2 MidAtlantic          7 Southwest
//   3 Southeast            8 Western
//   4 GreatLakes           9 Pacific
// This information is not online and I derived it by looking
// at post office maps. So the names may not be correct but it 20:
// is close enough for postal work.

try {
    zoneOfZipCode = Integer.parseInt(zipCode.substring(0, 1));
} catch(Exception e) {
    return vectorOfFulfillers; //passed in a malformed zip code
}
fulfillersTmp = Fulfiller.getByZone(zoneOfZipCode);
for (i = 0; i < fulfillersTmp.size(); i++)
   vectorOfFulfillers.addElement(fulfillersTmp.elementAt(i));

while (true) {
  keepGoing = false;

  // we may get a zone in the middle of the country so we need
  // to step away 1 zone at a time to make sure that we get the
  // fulfillers closest to this zone

  if (zoneOfZipCode + step <= Fulfiller.LAST_ZIP_ZONE) {
    fulfillersTmp = Fulfiller.getByZone(zoneOfZipCode + step);
    for (i = 0; i < fulfillersTmp.size(); i++)
     vectorOfFulfillers.addElement(fulfillersTmp.elementAt(i));
    keepGoing = true;
  }

  if (zoneOfZipCode - step >= Fulfiller.FIRST_ZIP_ZONE) {
    fulfillersTmp = Fulfiller.getByZone(zoneOfZipCode - step);
    for (i = 0; i < fulfillersTmp.size(); i++)
     vectorOfFulfillers.addElement(fulfillersTmp.elementAt(i));
    keepGoing = true;
  }

  if (keepGoing == true)
    step++;
  else
    break;
} // end while true

return vectorOfFulfillers;
}
```